Text-to-Speech: Adventures with Alex

Session 129

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

At the convention center, a movie will be placed here

Text-to-Speech: Adventures with Alex

Kim Silverman, Ph.D.
Principal Research Scientist
Manager, Spoken Language Technologies

Who Are We, What Do We Do?

- A team of research scientists and senior software engineers
- Part of Mac OS X
- Technologies
  - Text-to-speech synthesis
  - Speech recognition
  - Latent semantic mapping
  - Integration of these technologies across Apple products

Who Are We, What Do We Do?

- Text-to-speech synthesis

“Text-to-speech is the new black.”

David Pogue New York Times, March 18, 2009

What TTS Is in OS X?

- 24 different voices
  - Different technologies
  - Male, female, child, novelty voices
  - Highest quality: Alex

Fred

What TTS Is in OS X?

- Who is Alex?
  - Adult male
  - Highly natural articulation
  - Intonation tracks the topic structure of texts
  - Alex breathes!

What Can Be Done with Our TTS?

Accessibility

What Can Be Done with Our TTS?

Enhance the user interface

What Can Be Done with Our TTS?

The sole user interface

What Can Be Done with Our TTS?

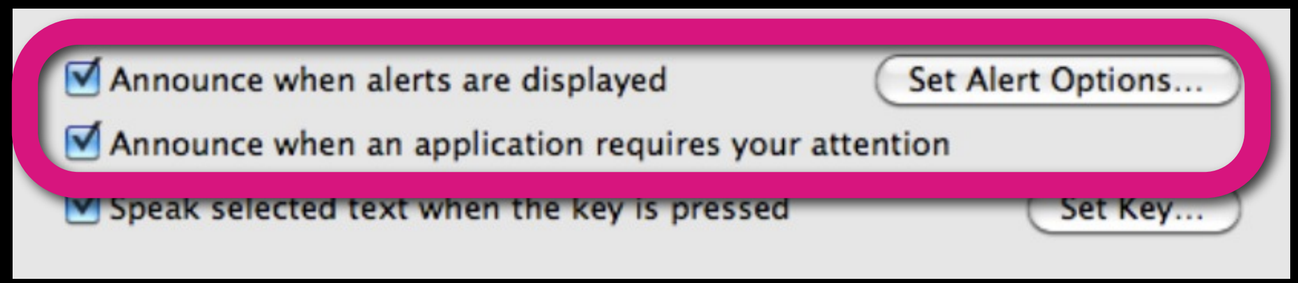
Broadcast media

What Can Be Done with Our TTS?

Mac OS X UI

What Can Be Done with Our TTS?

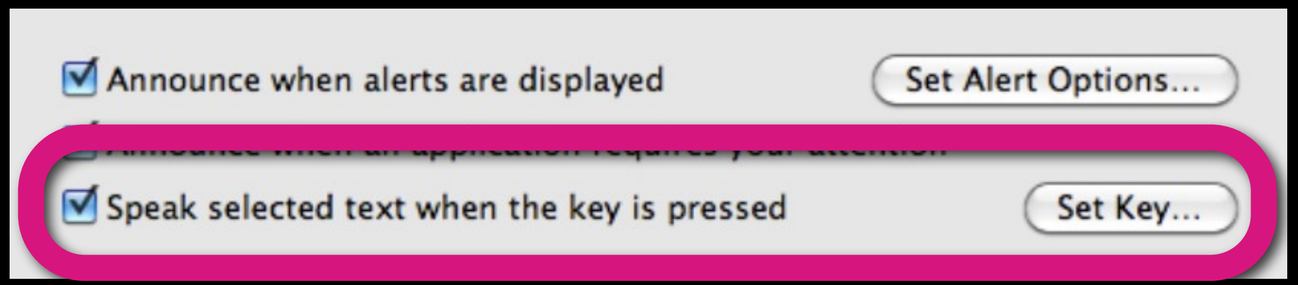
Mac OS X UI



- When not attending to the screen
- A background operation needs help
- K–12 education

What Can Be Done with Our TTS?

Mac OS X UI



- Reading PDFs, web pages, and email messages
  - Multitasking, "ageing eyes"
- Proofing messages before sending them
  - Saves embarrassment, kid's homework
- Language learners
  - How is that word pronounced?

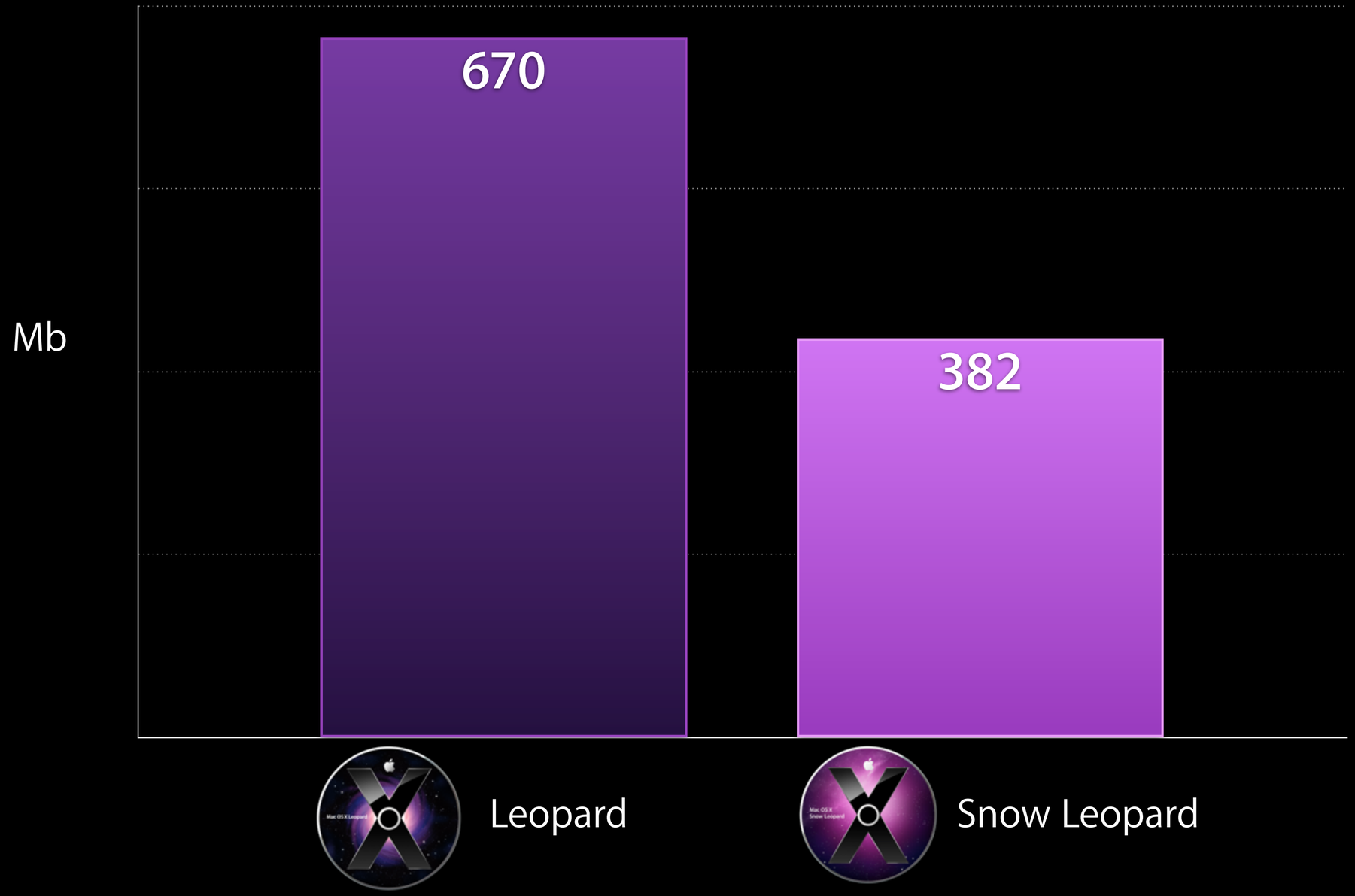
What Can Be Done with Our TTS?

Mac OS X UI

Announce or speak iChat messages

Email, RSS feeds into podcasts

Putting Alex on a Diet



Improved Speech Quality

- More natural intonation
  - Pitch more directly conveys the meaning
  - Smoother rhythm and phoneme durations
  - Better placement of pauses and phrases
- Significantly improved pronunciation
  - Names, performers, composers, actors
  - Technical terms
  - Ambiguous words
  - Classical music expressions
- Significant reduction of:
  - Dropouts, discontinuities, misarticulations, disfluencies, hesitations, glugs, pops, clicks, gurgles...

A Few Good Ideas…

- Expanded API functionality
  - Do more, with less work
- Output to Audio Units
  - Add rich audio effects
- Plug-ins!
  - Lots of new voices, dialects, languages

Create System

Keiko

Japanese

Cepstral LLC

Lawrence

English (UK)

Acapela Group

Peter

English (UK)

CereProc

Heather

Scottish English

Lots of voices!!!

...So, how do I get started?

Getting Started with Text-to-Speech

Kevin Aitken
Senior Software Engineer
Spoken Language Technologies

Text-to-Speech Programming Basics

- Same API for Apple and third-party synthesizers
- Voice determines language and synthesizer
- Accepts any string to speak, but special commands can be embedded
- Options for rate, volume, pitch, creating audio files, and more
- Naturally asynchronous

Accessing Text-to-Speech in Mac OS X

- AppleScript
- Command-line tool
- Cocoa
- C
- Ruby, Python, Lua

AppleScript

“say” command

- If an application can run an AppleScript, it can speak
- Integrates well with speech recognition
- Allows specifying the voice, saving audio to a file, and more
- See “StandardAdditions.osax” dictionary for documentation

AppleScript

“say” command

- If an application can run an AppleScript, it can speak
- Integrates well with speech recognition
- Allows specifying the voice, saving audio to a file, and more
- See “StandardAdditions.osax” dictionary for documentation

```
say "Now compressing image files."

say "You have mail from your boss!" using "Ralph"

say "More drivel from yours truly." saving to "MyBlogPodcast.aiff"
```

AppleScript

“say” command

- If an application can run an AppleScript, it can speak
- Integrates well with speech recognition
- Allows specifying the voice, saving audio to a file, and more
- See “StandardAdditions.osax” dictionary for documentation

```
say "Now compressing image files."

say "You have mail from your boss!" using "Ralph"

say "More drivel from yours truly." saving to "MyBlogPodcast.aiff"
```

AppleScript

“say” command

- If an application can run an AppleScript, it can speak
- Integrates well with speech recognition
- Allows specifying the voice, saving audio to a file, and more
- See “StandardAdditions.osax” dictionary for documentation

```
say "Now compressing image files."

say "You have mail from your boss!" using "Ralph"

say "More drivel from yours truly." saving to "MyBlogPodcast.aiff"
```

Command-Line Tool

“say” command

- Waits until speaking is finished
- Allows specifying the voice and reading from a file
- Flexible support for saving audio to a file
- “man say” for documentation

Command-Line Tool

“say” command

- Waits until speaking is finished
- Allows specifying the voice and reading from a file
- Flexible support for saving audio to a file
- “man say” for documentation

```
say "It's 12:30, time for lunch!"

say -v Vicki "Your status report is due."

say -f ATaleOfTwoCites.txt -o ListenWhileIDrive.aiff

xcodebuild; say "Build complete!"
```

Command-Line Tool

“say” command

- Waits until speaking is finished
- Allows specifying the voice and reading from a file
- Flexible support for saving audio to a file
- “man say” for documentation

```
say "It's 12:30, time for lunch!"

say -v Vicki "Your status report is due."

say -f ATaleOfTwoCites.txt -o ListenWhileIDrive.aiff

xcodebuild; say "Build complete!"
```

Command-Line Tool

“say” command

- Waits until speaking is finished
- Allows specifying the voice and reading from a file
- Flexible support for saving audio to a file
- “man say” for documentation

```
say "It's 12:30, time for lunch!"

say -v Vicki "Your status report is due."

say -f ATaleOfTwoCites.txt -o ListenWhileIDrive.aiff

xcodebuild; say "Build complete!"
```

Command-Line Tool

“say” command

- Waits until speaking is finished
- Allows specifying the voice and reading from a file
- Flexible support for saving audio to a file
- “man say” for documentation

```
say "It's 12:30, time for lunch!"

say -v Vicki "Your status report is due."

say -f ATaleOfTwoCites.txt -o ListenWhileIDrive.aiff

xcodebuild; say "Build complete!"
```

Cocoa and CF-Based APIs

- Full access to synthesizers
- Direct setting/getting of synthesizer properties
- Support for notification types, such as
  - Speaking completed
  - Word and phoneme
- Access to list of available voices and their properties
  - Name, age, gender, locale
- Routine for text-to-phoneme conversion
- Support for custom dictionaries

NSSpeechSynthesizer

Getting started

- Link with AppKit.framework or Cocoa.framework
- Use a RunLoop to receive delegate messages

NSSpeechSynthesizer

Making it speak

- Create an instance of NSSpeechSynthesizer

```
NSSpeechSynthesizer * synthesizer =
                                [NSSpeechSynthesizer new];

[synthesizer setDelegate:self];

[synthesizer startSpeakingString:@"Don't object to my objects!"];
```

NSSpeechSynthesizer

Making it speak

- Set the delegate

```
NSSpeechSynthesizer * synthesizer =
                                [NSSpeechSynthesizer new];

[synthesizer setDelegate:self];

[synthesizer startSpeakingString:@"Don't object to my objects!"];
```

NSSpeechSynthesizer

Making it speak

- Begin speaking

```
NSSpeechSynthesizer * synthesizer =
                                [NSSpeechSynthesizer new];

[synthesizer setDelegate:self];

[synthesizer startSpeakingString:@"Don't object to my objects!"];
```

NSSpeechSynthesizer

Handling delegate messages

- Message when speaking has finished

```
- (void)speechSynthesizer:(NSSpeechSynthesizer *)sender
        didFinishSpeaking:(BOOL)finishedSpeaking {

    // Speaking has finished, update the user interface.

}
```

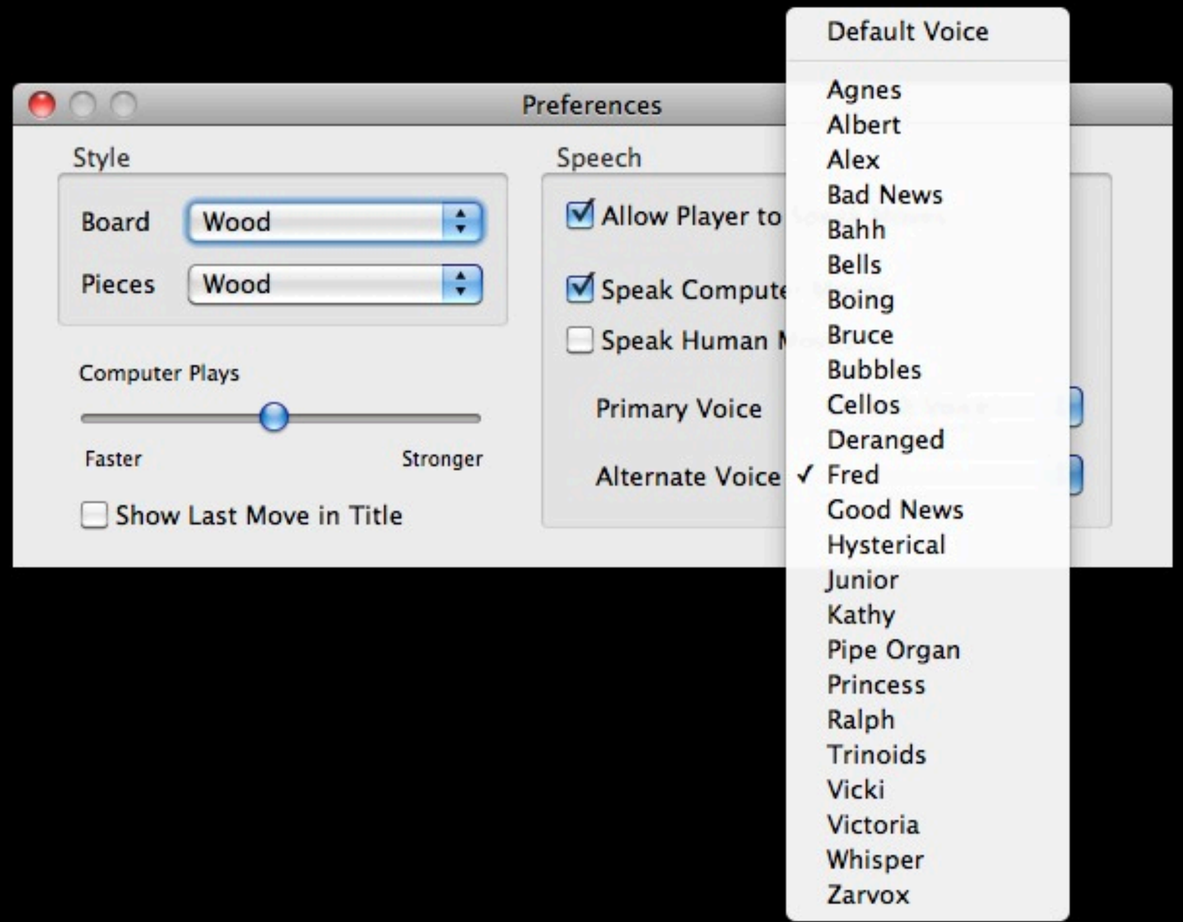
NSSpeechSynthesizer

Managing voices

- Provides a list of available voices
- Properties of voices include name, gender, age, locale

NSSpeechSynthesizer

Managing voices



NSSpeechSynthesizer

Managing voices

- Getting the list of available voices

```
for (NSString * aVoice in [NSSpeechSynthesizer availableVoices]) {

 NSDictionary *  voiceInfo =
                    [NSSpeechSynthesizer attributesForVoice:aVoice];

 NSString * voiceName =
                    [voiceInfo objectForKey:NSVoiceName];

 // Add voice name to menu

}
```

NSSpeechSynthesizer

Managing voices

- Getting information about a voice

```
for (NSString * aVoice in [NSSpeechSynthesizer availableVoices]) {

  NSDictionary *  voiceInfo =
                    [NSSpeechSynthesizer attributesForVoice:aVoice];

  NSString * voiceName =
                    [voiceInfo objectForKey:NSVoiceName];

  // Add voice name to menu

}
```

NSSpeechSynthesizer

Managing voices

- Getting the voice name

```
for (NSString * aVoice in [NSSpeechSynthesizer availableVoices]) {

  NSDictionary *  voiceInfo =
                    [NSSpeechSynthesizer attributesForVoice:aVoice];

  NSString * voiceName =
                    [voiceInfo objectForKey:NSVoiceName];

  // Add voice name to menu

}
```

CF-Based Speech Synthesis API

Getting started

- Link with ApplicationServices.framework
- Using a RunLoop while speaking is a good idea

CF-Based Speech Synthesis API

Making it speak

- Create a speech channel

```
SpeechChannel * chan;
err = NewSpeechChannel(NULL, &chan);

CFNumber * callback =
        CFNumberCreate(NULL, kCFNumberLongType, HighlightSpokenWord);
err = SetSpeechProperty(chan, kSpeechWordCFCallBack, callback);

err = SpeakCFString(chan, CFSTR("See Spot Code!"), NULL);
```

CF-Based Speech Synthesis API

Making it speak

- Place word callback proc pointer in a CFNumberRef

```
SpeechChannel * chan;
err = NewSpeechChannel(NULL, &chan);

CFNumber * callback =
        CFNumberCreate(NULL, kCFNumberLongType, HighlightSpokenWord);
err = SetSpeechProperty(chan, kSpeechWordCFCallBack, callback);

err = SpeakCFString(chan, CFSTR("See Spot Code!"), NULL);
```

CF-Based Speech Synthesis API

Making it speak

- Set the word callback proc

```
SpeechChannel * chan;
err = NewSpeechChannel(NULL, &chan);

CFNumber * callback =
        CFNumberCreate(NULL, kCFNumberLongType, HighlightSpokenWord);
err = SetSpeechProperty(chan, kSpeechWordCFCallBack, callback);

err = SpeakCFString(chan, CFSTR("See Spot Code!"), NULL);
```

CF-Based Speech Synthesis API

Making it speak

- Begin speaking

```
SpeechChannel * chan;
err = NewSpeechChannel(NULL, &chan);

CFNumber * callback =
        CFNumberCreate(NULL, kCFNumberLongType, HighlightSpokenWord);
err = SetSpeechProperty(chan, kSpeechWordCFCallBack, callback);

err = SpeakCFString(chan, CFSTR("See Spot Code!"), NULL);
```

CF-Based Speech Synthesis API

Handling callbacks

- Callback is called before speaking the word

```
void HighlightSpokenWord( SpeechChannel chan,
                          SRefCon refCon,
                          CFStringRef aString,
                          CFRange wordRange)
{
  // Highlight the word.
}
```

Ruby, Python, Lua

Ruby example

Ruby, Python, Lua

Ruby example

- Create an instance of NSSpeechSynthesizer

```
#!/usr/bin/ruby
require 'osx/cocoa'

synthesizer = OSX::NSSpeechSynthesizer.new
synthesizer.startSpeakingString "Calling Cocoa from Ruby is easy."

while synthesizer.isSpeaking
    OSX::NSRunLoop.mainRunLoop.runUntilDate(
            OSX::NSDate.dateWithTimeIntervalSinceNow 1.0)
end
```

Ruby, Python, Lua

Ruby example

- Begin speaking

```
#!/usr/bin/ruby
require 'osx/cocoa'

synthesizer = OSX::NSSpeechSynthesizer.new
synthesizer.startSpeakingString "Calling Cocoa from Ruby is easy."

while synthesizer.isSpeaking
    OSX::NSRunLoop.mainRunLoop.runUntilDate(
            OSX::NSDate.dateWithTimeIntervalSinceNow 1.0)
end
```

Ruby, Python, Lua

Ruby example

- Loop until speaking has finished

```
#!/usr/bin/ruby
require 'osx/cocoa'

synthesizer = OSX::NSSpeechSynthesizer.new
synthesizer.startSpeakingString "Calling Cocoa from Ruby is easy."

while synthesizer.isSpeaking
    OSX::NSRunLoop.mainRunLoop.runUntilDate(
            OSX::NSDate.dateWithTimeIntervalSinceNow 1.0)
end
```

On-the-Fly Voice/Synthesizer Loading

- New in Snow Leopard
- APIs for using voices and synthesizers outside of the standard directories
- Allows applications to bundle third-party synthesizers and voices
- Simplifies licensing issues

Bundling a Synthesizer into Your Application

Start with your application

YourApp

Bundling a Synthesizer into Your Application

License a synthesizer and voice

YourApp

xyz.SpeechSynthesizer

xyzVoice.SpeechVoice

Bundling a Synthesizer into Your Application

Embed the synthesizer and voice

YourApp

xyz.SpeechSynthesizer

xyzVoice.SpeechVoice

Resources

Bundling a Synthesizer into Your Application

Register the synthesizer and voice at launch

```
.
.
.
SpeechSynthesisRegisterModuleURL( xyzSynthesizerURL );
SpeechSynthesisRegisterModuleURL( xyzVoiceURL );
.
.
.
```

Demo

Bundled Acapela synthesizer

Getting Started

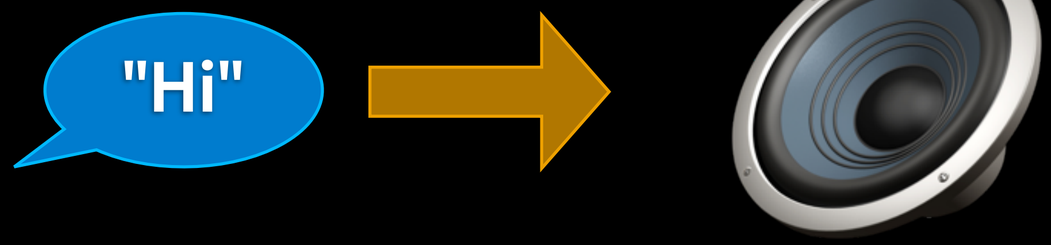
Examples
/Developer/Examples/Speech/

Documentation
developer.apple.com

Audio Output

Matthias Neeracher, Dr. Sc. Techn.
Senior Software Engineer
Spoken Language Technologies

Say It!

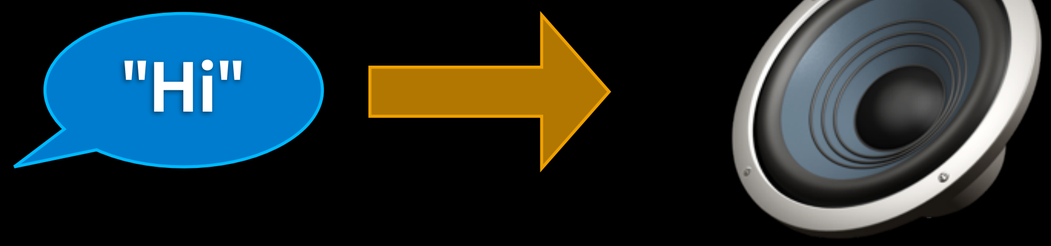


- Speak to default sound output device

```
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say "Hi"
```

Say It!

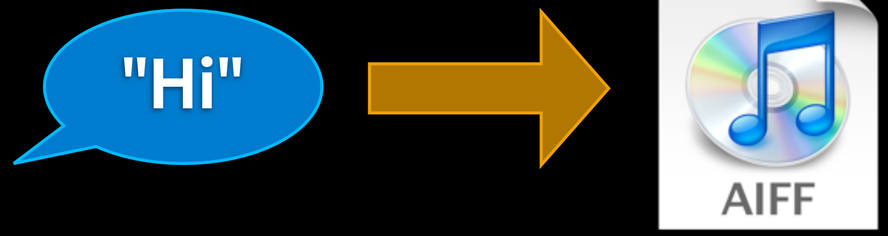


- Speak to default sound output device

```
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say "Hi"
```

Save It!

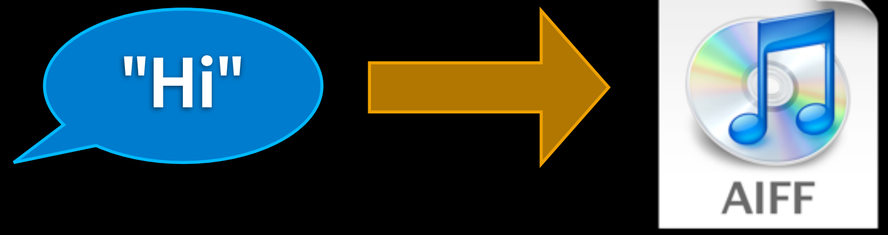


- Speak to AIFF file

```
SetSpeechProperty(channel, kSpeechOutputToFileURLProperty, hiURL);
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say -o hi.aiff "Hi"
```

Save It!

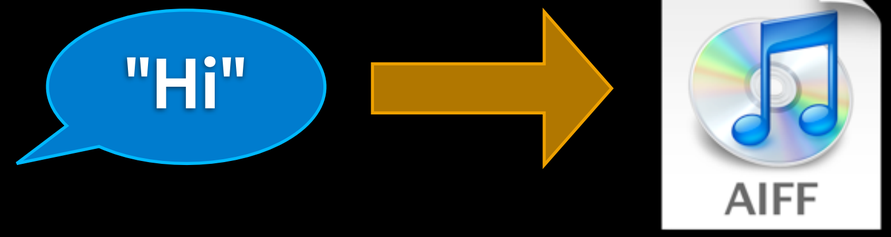


- Speak to AIFF file

```
SetSpeechProperty(channel, kSpeechOutputToFileURLProperty, hiURL);
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say -o hi.aiff "Hi"
```

Save It!

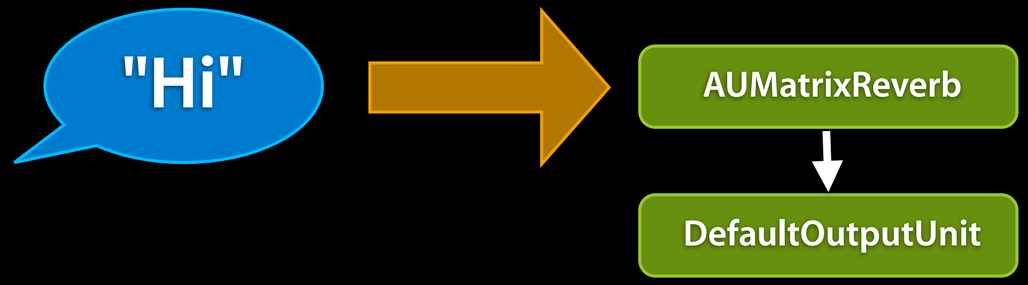


- Speak to AIFF file

```
SetSpeechProperty(channel, kSpeechOutputToFileURLProperty, hiURL);
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say -o hi.aiff "Hi"
```

Process It!



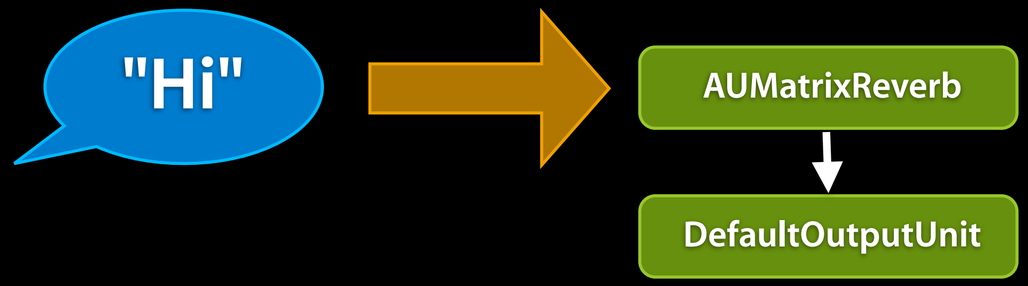
- Feed into Audio Unit(s)

```
AudioComponentDescription cd = {
   kAudioUnitType_Generator, kAudioUnitSubType_SpeechSynthesis,
   kAudio
AudioComp                                                  &speechUnit);
AudioUnit
   speech
   kAudio
SpeakCFSt


say -n : -f Moby_Dick.txt
```

Track 29
Chattanooga Choo Choo

Process It!

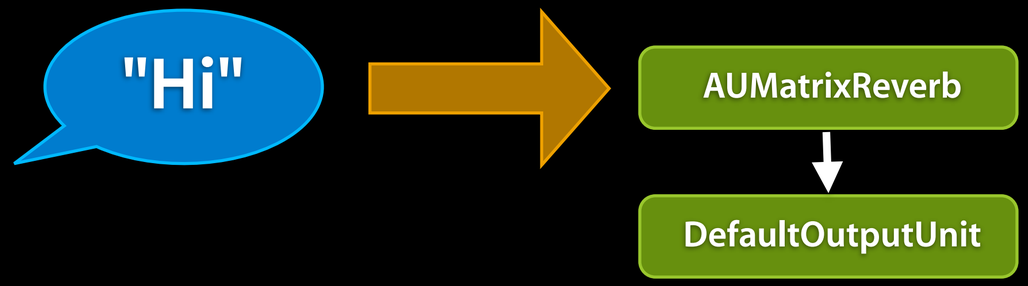


- Feed into Audio Unit(s)

```
AudioComponentDescription cd = {
   kAudioUnitType_Generator, kAudioUnitSubType_SpeechSynthesis,
   kAudioUnitManufacturer_Apple, 0, 0};
AudioComponentInstanceNew(AudioComponentFindNext(NULL, &cd), &speechUnit);
AudioUnitGetProperty(
   speechUnit, kAudioUnitProperty_SpeechChannel,
   kAudioUnitScope_Global, 0, &channel, &sz);
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -n : -f Moby_Dick.txt
```

Process It!

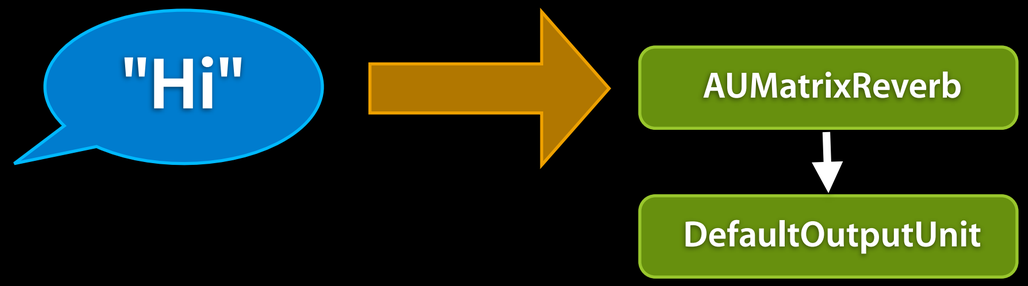


- Feed into Audio Unit(s)

```
AudioComponentDescription cd = {
   kAudioUnitType_Generator, kAudioUnitSubType_SpeechSynthesis,
   kAudioUnitManufacturer_Apple, 0, 0};
AudioComponentInstanceNew(AudioComponentFindNext(NULL, &cd), &speechUnit);
AudioUnitGetProperty(
   speechUnit, kAudioUnitProperty_SpeechChannel,
   kAudioUnitScope_Global, 0, &channel, &sz);
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -n : -f Moby_Dick.txt
```

Process It!

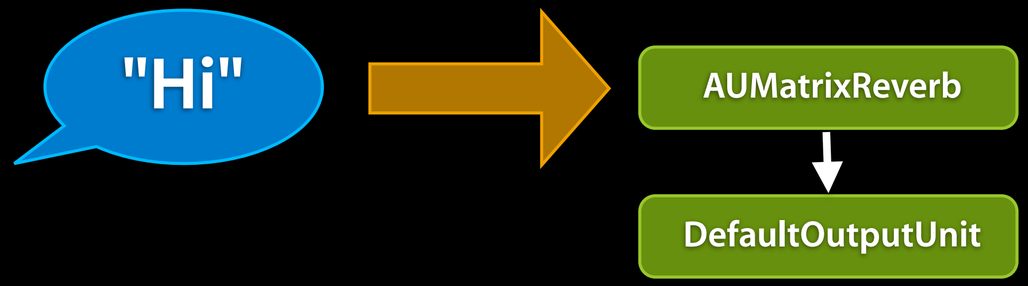


- Feed into Audio Unit(s)

```
AudioComponentDescription cd = {
   kAudioUnitType_Generator, kAudioUnitSubType_SpeechSynthesis,
   kAudioUnitManufacturer_Apple, 0, 0};
AudioComponentInstanceNew(AudioComponentFindNext(NULL, &cd), &speechUnit);
AudioUnitGetProperty(
   speechUnit, kAudioUnitProperty_SpeechChannel,
   kAudioUnitScope_Global, 0, &channel, &sz);
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -n : -f Moby_Dick.txt
```

Process It!



- Feed into Audio Unit(s)

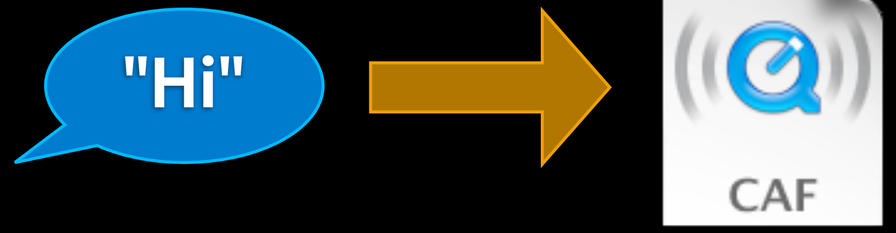
```
AudioComponentDescription cd = {
   kAudioUnitType_Generator, kAudioUnitSubType_SpeechSynthesis,
   kAudioUnitManufacturer_Apple, 0, 0};
AudioComponentInstanceNew(AudioComponentFindNext(NULL, &cd), &speechUnit);
AudioUnitGetProperty(
   speechUnit, kAudioUnitProperty_SpeechChannel,
   kAudioUnitScope_Global, 0, &channel, &sz);
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -n : -f Moby_Dick.txt
```

Demo

Speech with Audio Units

Encode It!

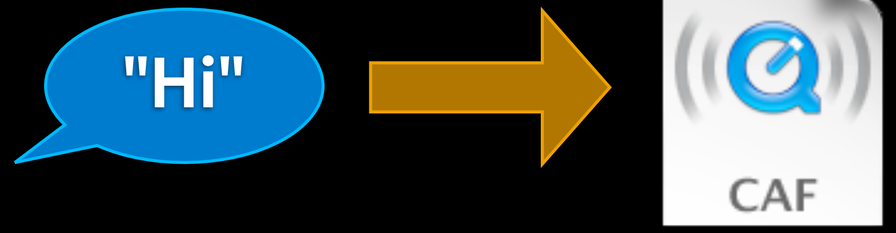


- Speak to ExtAudioFile

```
ExtAudioFileCreateWithURL(url, fileFormat, &dataFormat, NULL,
        kAudioFileFlags_EraseFile, &audioFile);
SetSpeechProperty(channel, kSpeechOutputToExtAudioFileProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &audioFile));
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -o hi.caf --data-format=alac "Hi"
```

Encode It!

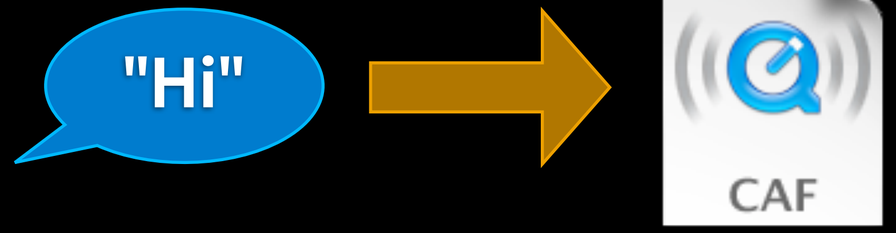


- Speak to ExtAudioFile

```
ExtAudioFileCreateWithURL(url, fileFormat, &dataFormat, NULL,
        kAudioFileFlags_EraseFile, &audioFile);
SetSpeechProperty(channel, kSpeechOutputToExtAudioFileProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &audioFile));
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say -o hi.caf --data-format=alac "Hi"
```

Encode It!

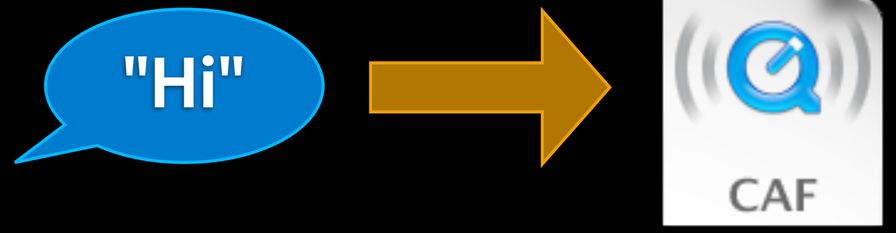


- Speak to ExtAudioFile

```
ExtAudioFileCreateWithURL(url, fileFormat, &dataFormat, NULL,
        kAudioFileFlags_EraseFile, &audioFile);
SetSpeechProperty(channel, kSpeechOutputToExtAudioFileProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &audioFile));
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -o hi.caf --data-format=alac "Hi"
```

Encode It!

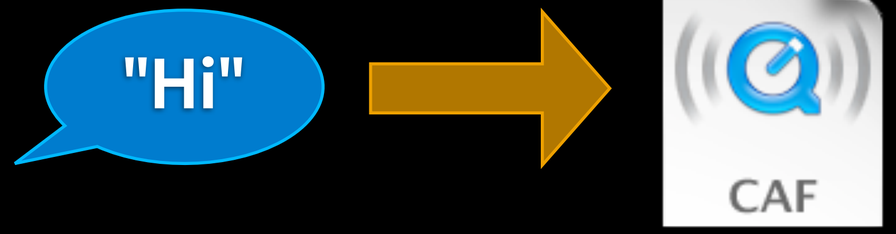


- Speak to ExtAudioFile

```
ExtAudioFileCreateWithURL(url, fileFormat, &dataFormat, NULL,
        kAudioFileFlags_EraseFile, &audioFile);
SetSpeechProperty(channel, kSpeechOutputToExtAudioFileProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &audioFile));
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

```
say -o hi.caf --data-format=alac "Hi"
```

Encode It!



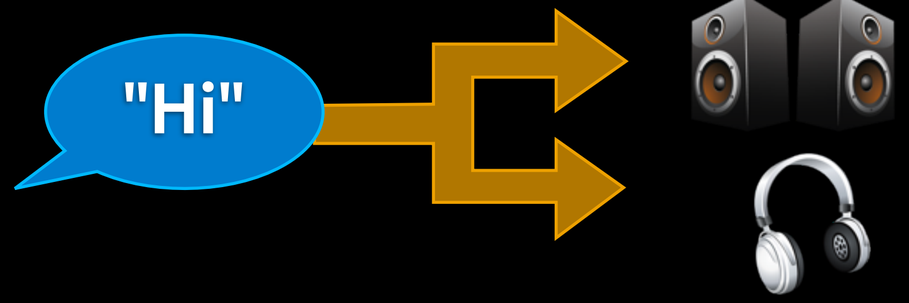
- Speak to ExtAudioFile

```
ExtAudioFileCreateWithURL(url, fileFormat, &dataFormat, NULL,
        kAudioFileFlags_EraseFile, &audioFile);
SetSpeechProperty(channel, kSpeechOutputToExtAudioFileProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &audioFile));
SpeakCFString(channel, CFSTR("Hi"), NULL);


say -o hi.caf --data-format=alac "Hi"
```

- Does not need to be a file!

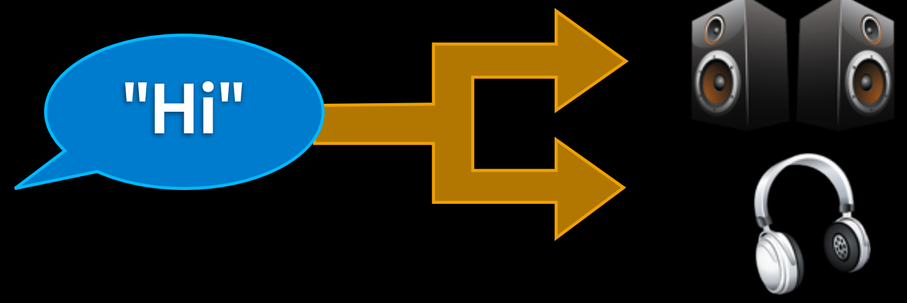
Redirect It!



- Speak to any audio device, instead of always to default

```
AudioDeviceID phatSpeakers = ...;
SetSpeechProperty(channel, kSpeechOutputToAudioDeviceProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &phatSpeakers));
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

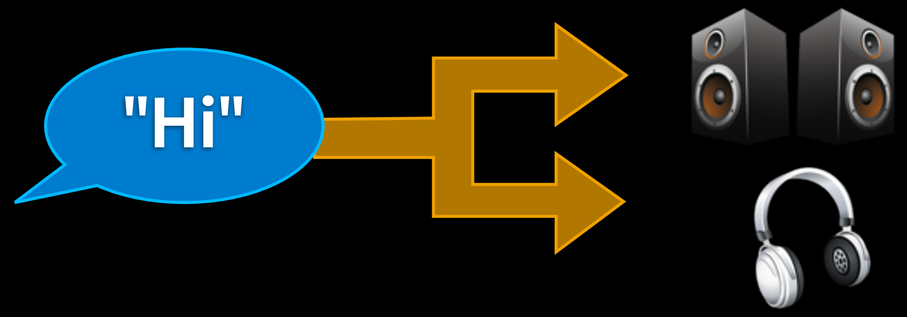
Redirect It!



- Speak to any audio device, instead of always to default

```
AudioDeviceID phatSpeakers = ...;
SetSpeechProperty(channel, kSpeechOutputToAudioDeviceProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &phatSpeakers));
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

Redirect It!



- Speak to any audio device, instead of always to default

```
AudioDeviceID phatSpeakers = ...;
SetSpeechProperty(channel, kSpeechOutputToAudioDeviceProperty,
        CFNumberCreate(NULL, kCFNumberLongType, &phatSpeakers));
SpeakCFString(channel, CFSTR("Hi"), NULL);
```

Availability

MacinTalk

- Speak to default audio device: 10.0 (Cheetah)
- Speak to AIFF file: 10.2 (Jaguar)
- Speak to Audio Units: 10.4 (Tiger)
- Speak to ExtAudioFile: 10.6 (SnowLeopard)
- Speak to AudioDevice: 10.6 (SnowLeopard)

Third-Party Synthesizers

- Speak to default audio device universally available
- Speak to AIFF file generally available
- Other methods not widely adopted yet

Where to Go from Here

Kim Silverman, Ph.D.
Principal Research Scientist
Manager, Spoken Language Technologies

Three Steps

Step one

- There is no step one
  - Your app already speaks!
    - VoiceOver
    - Talking Alerts
    - Announce when an application requires your attention
    - Speak any selected text
    - Speakable Items: Front window commands

Three Steps

Step one

- There is no step one—your app already speaks
  - So make sure it sounds good!

Step two

- Differentiate your app
  - Use speech in ways appropriate to your user interaction

Where Should I Use Speech?

Asynchronous notifications

- Something happens outside of a user's immediate control
  - *"Time's up!"*
  - *"Data backup is complete"*
  - *"Urgent email"*
  - *"Enemy planes at twelve o'clock"*

Where Should I Use Speech?

Asynchronous notifications

Feedback to the user

- *“Game over”*
- When changing weapons

First Person Shooters

Where Should I Use Speech?

Asynchronous notifications

Feedback to the user

Confirm user actions

- *“Sending a check to AT&T for $60”*
- Read back list information a user has entered
  - Names, budget, timetables, etc.

Where Should I Use Speech?

Asynchronous notifications

Feedback to the user

Confirm user actions

Add speech in ways specific to your application

- A script or play
  - Read different characters in different voices
  - Do not read back acting directions or blocking cues
  - Allow user to select one character to NOT read, for rehearsal purposes

Three Steps

Step one

- There is no step one—your app already speaks
  - So make sure it sounds good!

Step two

- Differentiate your app
  - Use speech in ways appropriate to your user interaction

Step three

- Customize!

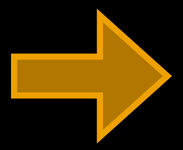
Customize What???

- What words are spoken
  - UESC .44 Magnum Mega Class A1. *“Magnum”*
- How words are pronounced
  - *Don’t say Raiden, instead pronounce it: [[inpt phon]] r1EYdIXn [[inpt text]]*
- Speaking rate
  - *“[[rate 150]] Sometimes the default speaking rate might make the speech sound somewhat tedious and pedantic”*
  - The more predictable the material, the faster you can speak it
- Intonation
  - Use [[emph -]] to remove the pitch accent in compound nouns

Compound Nouns

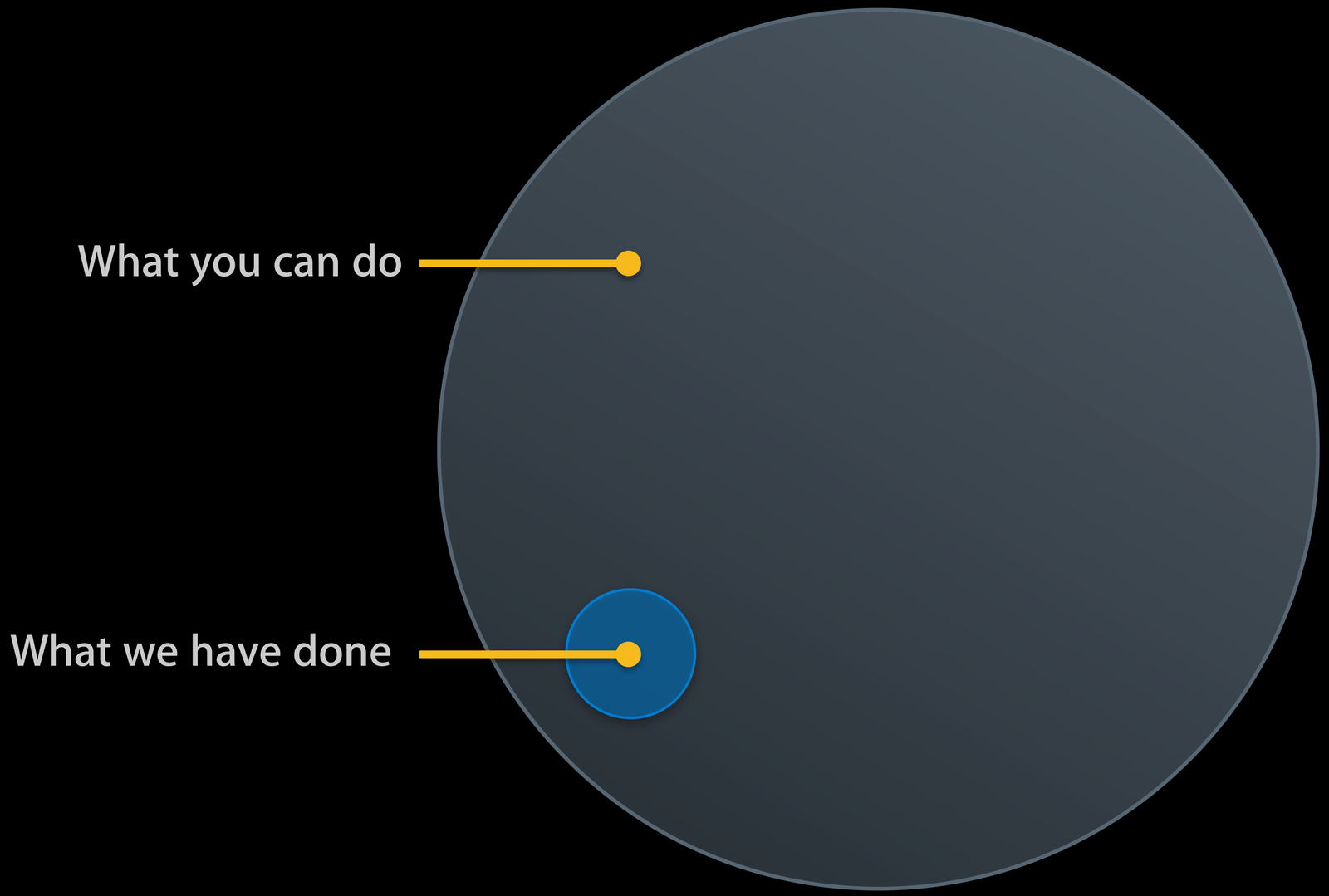
Flame Thrower	Flame [[emph -]] Thrower
Plasma Grenade	Plasma [[emph -]] Grenade
Gravity Hammer	Gravity [[emph -]] Hammer
Project List	Project [[emph -]] List
Street View	Street [[emph -]] view

Compound Nouns



Flame Thrower	Flame [[emph -]] Thrower
Plasma Grenade	Plasma [[emph -]] Grenade
Project List	Project [[emph -]] List
Street View	Street [[emph -]] view

Things That Can Be Done with Speech in Your App



Now you're talking!!!

Now you're talking!!!

...any questions?

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09